NOAO E2E Integrated Data Cache Initiative Using iRODS

Irene Barg, Derec Scott, Erik Timmermann
National Optical Astronomy Observatory, Science Data Management, Tucson, AZ 85719

Introduction

SRB was nearing its end of life status when the first release of iRODS appeared in 2008. At the same time, the Science Data Management Operations (SDM) group were making plans to move away from the Apple XSAN architecture for our mass storage system (MSS). The two NOAO data centers, Tucson, Arizona and La Serena, Chile, provide the 'front end' to the MSS for all services that required access to the Archive. These frontend systems were Apple XServers running XSan client software. It was getting increasingly difficult to maintain both Mac OS X upgrades and the SRB due to compatibility issues. It was time to not only retire SRB but to migrate our MSS front-end systems to Linux.

Why iRODS?

The SRB had been a key component of the DCI since it's inception (August 2004). Since iRODS was the successor to SRB, we decided to take a look at iRODS. After downloading and installing iRODS in our test environment, it felt familiar, and the functions we had grown to rely on were still there.

Ease of Migration

iRODS migration requirements:

Migration requirements	iRODS	Comment
Retain existing physical and logical resources	yes	
Retain collection hierarchy	yes	
Retain existing mdasCollectionHome	yes	
Retain existing groups (noao, smarts)	no	Implement manually
Retain access control (ACL)	no	Implement manually
Retain user metadata (md5sum)	no	iRODS checksum is md5

The three requirements not met had acceptable workaround. Discussions with two members of the iRODS team suggested that the best migration path would be as follows:

1. Recreate the physical and logical resources in iRODS;
2. Bulk register the existing holdings;
3. Recreate your institutional groups + users;
4. Recreate the access control.

There were pros/cons to this approach. The pros were:

- Clean installation;
- Allowed us to make changes;
- Could be implemented in parallel to live system.

The cons were:

- Time consuming.

The ability to run an iRODS system in parallel to the SRB in order to implement a phased migration mitigated the 'time consuming' disadvantage.

Phased Migration from SRB to iRODS

The migration to iRODS was implemented in phases. There were three drivers to a phased approach:

1. We had to keep the current SRB-MCAT system running to support the production Archive.
2. We needed all pre-existing MCAT metadata available at the time we switched to iRODS;
3. We needed to time the transition to full iRODS with an Archive release that supported iRODS.

A brief description of this hybrid system follows:

1. Two new iRODS scripts written to take advantage of bundled transfers;
2. A modified version of the DCI client was written to register files in both SRB and iRODS;
3. A process to migrate MCAT-to-ICAT was implemented.

Phasing the migration this way allowed us to make the switch with the least amount of downtime to the production system as possible.

Reduction in code

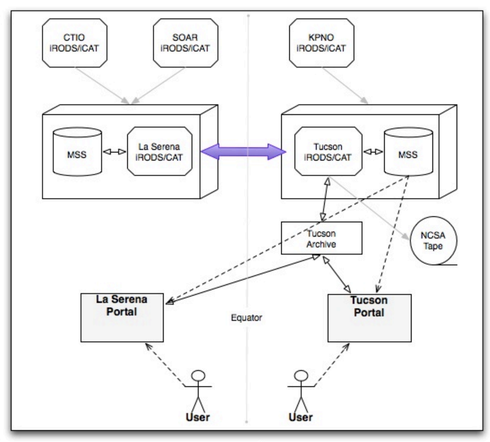
The original DCI code used the Perl API for SRB. This code needed to be re-written to utilize the iRODS implementation. However, this was a good thing, not bad. At the iRODS core is a Rule Engine that interprets the Rules to decide how the system is to respond to various requests. Micro services are small, well-defined procedures/functions that perform a certain task. These micro-services and the rule engine interact to provide you with control over what happens when one performs a macro-level functionality. Functionality that was originally implemented in the API, are now implemented using a set of rules. For example we can set the access control (ACL) for each file using a set of pre-defined rules.

Improved Performance

Improved performance was achieved in many ways:

1. **Simpler code** - moving repetitive tasks at the rule and micro-service level that get compiled into the iRODS server code results in faster run-time application. We replaced the SRB Perl API code with more efficient icommands. With finer control being done at the rule level, most of the functionality of the previous DCI code is now implemented with 2-3 simple icommands.
2. **Bundle file operations** - bundling many small files into a large 350MB tar file then using iRODS *iget* to transfer the tar bundle improved file transfer efficiency overall by 30% .
3. **Improved file transfer performance** - was achieved by multi-threaded, concurrent, parallel TCP/IP connections.

The current iDCI data flow is illustrated in Figure 1.



Summary

iRODS is easy to install and configure. iRODS rule base engine and use of micro-services gives the end user greater flexibility and control over functionality of your file repository system. We have just started learning the capabilities of iRODS. We created an *auto-retire* program that uses a *md5checker to* verify a night of data and if successful, adds a comment to the iRODS root collection for the night. ICAT collections which have the verified comment and are greater than 9 weeks old are purged from the mountain caches automatically.

Acknowledgements

iRODS™, the Integrated Rule-Oriented Data System, is a data grid software system developed by the Data Intensive Cyber Environments (DICE) research group (developers of the SRB, the Storage Resource Broker), and collaborators (http://www.irods.org). We wish to thank the DICE team for their support and feedback and for providing such a fine product as iRODS.